In the specification:

Please replace paragraph [0008] with the following amended paragraph:

[0008] Figure 1 of the drawings shows an optimizing compiler 10 in which embodiments of the invention may be practiced. The optimizing compiler 10 includes a lexical analyzer 12 which takes a source program and breaks it up to meaningful units called tokens. A syntax analyzer 14 determines the structure of the program and of the individual statements therein by grouping the tokens into grammatical phrases which are then checked by a semantic analyzer 16 for semantic errors. The compiler 10 further includes an intermediate code generator 18 which generates an intermediate program representation of the source program in an intermediate language. A code optimizer 20 attempts to optimize[[r]] the program representation. The final phase of the compiler 10 is carried out by a code generator 22 which generates target comprising machine or assembly code.

Please replace paragraph [0010] with the following amended paragraph:

[0010] The present invention permits value analysis problems to be solved over large input programs without excessive time or space penalties. In particular, program analysis according to embodiments of the invention, includes constructing dependent flow graphs in which the number of edges in the dependence flow graph is linear [[in]] to the number of nodes in the graph, i.e. there are a constant number of edges per node. Embodiments of the invention make use of an equivalence class-based-class-based alias analysis of the intermediate language program to create dependence flow graphs which have the property that the edge cardinality is independent of the definition-use structure of the program being analyzed. An equivalence class is a class of overlapping memory accesses.

Please replace paragraphs [0017]-[0018] with the following amended paragraphs:

[0017] Figure 4 of the drawings shows an algorithm, in pseudocode, for performing to perform a flow insensitive program analysis, in accordance with one embodiment of the invention. Referring to the algorithm, A is equal to the number of aliases associated with

Docket No.: 42P12907
Application No.: 09/976,313   2

the GETs and PUTs of the program. G is a dependence flow graph defined in accordance with the above method and Q is a set of nodes of G, which is initially empty. The algorithm assigns an abstract value to each alias in the dependence flow graph. It is assumed that the abstract value from a joint complete partial order, and that for two abstract values V_1 and V_2, the expression LE (V_1, V_2) returns true if V_1 is less than or equal to V_2 in the partial order. The expression JOIN (V_1, V_2) returns the JOIN of V_1 and V_2 in the partial order. E1 is an expression in the program and Eval (E1) returns the value of E1. For each memory alias, M, the expression InitialValue (M) returns an abstract value that is a safe approximation of the initial contents of the storage location (s) represented by M.

[0018] Referring to **Figure 5** of the drawings reference numeral 100 generally indicates hardware for performing <u>program analysis</u> term rewriting in accordance with the invention. The hardware 100 includes a memory 104, which may represent one or more physical memory devices, which may include any type of random access memory (RAM) read only memory (ROM) (which may be programmable), flash memory, non-volatile mass storage device, or a combination of such memory devices. The memory 104 is connected via a system bus 112 to a processor 102. The memory 104 includes instructions 106 which when executed by the processor 102 cause the processor to perform the methodology of the invention as discussed above. Additionally the system 100 includes a disk drive 108 and a CD ROM drive 110 each of which is coupled to a peripheral-device and user-interface 114 via bus 112. Processor 102, memory 104, disk drive 108 and CD ROM 110 are generally known in the art. Peripheral-device and user-interface 114 provides an interface between system bus 112 and various components connected to a peripheral bus 116 as well as to user interface components, such as display, mouse and other user interface devices. A network interface 118 is coupled to peripheral bus 116 and provides network connectivity to system 100.